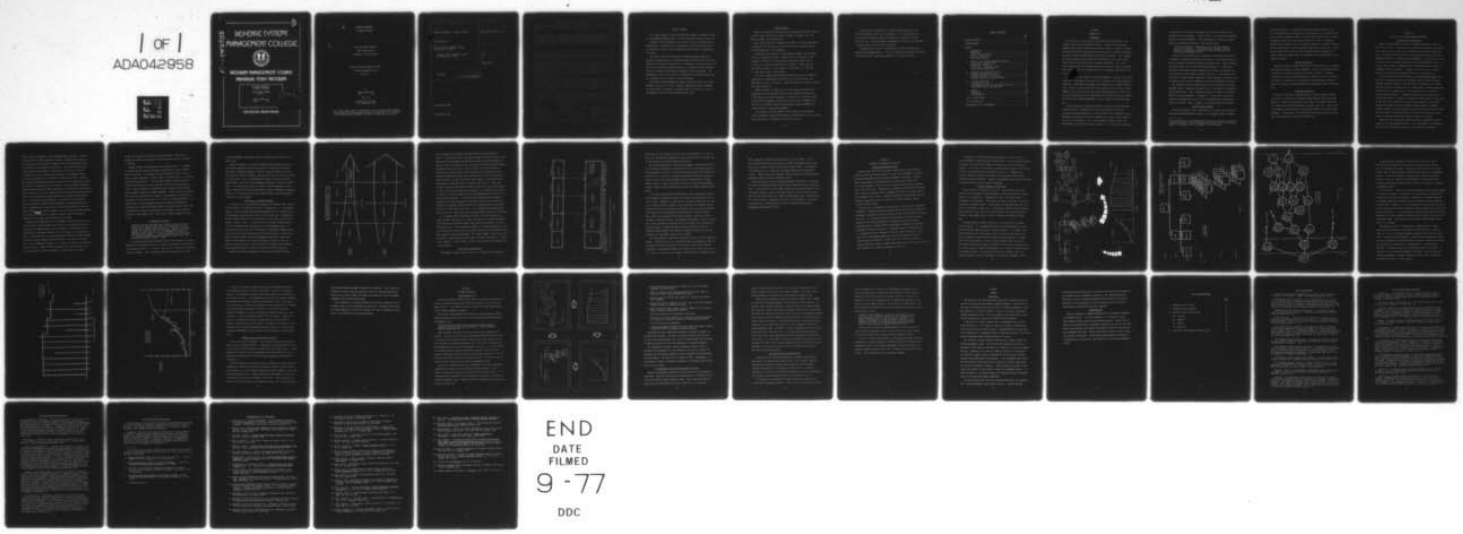
AD-A042 958 DEFENSE SYSTEMS MANAGEMENT COLL FORT BELVOIR VA F/G 5/1
SOFTWARE MANAGEMENT: A DYNAMIC APPROACH.(U)
MAY 77 E H ELY

UNCLASSIFIED NL

| OF |
ADA042958



END
DATE FILMED
9-77
DDC

DEFENSE SYSTEMS MANAGEMENT COLLEGE

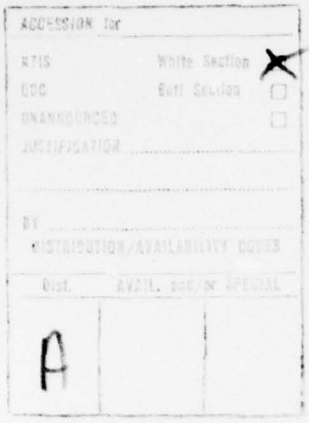
PROGRAM MANAGEMENT COURSE
INDIVIDUAL STUDY PROGRAM

SOFTWARE MANAGEMENT:
A DYNAMIC APPROACH

STUDY PROJECT REPORT
PMC 77-1

Edward Hudson Ely
MAJ USA

FORT BELVOIR, VIRGINIA 22060



SOFTWARE MANAGEMENT:

A DYNAMIC APPROACH

Individual Study Program

Study Project Report

Prepared as a Formal Report

Defense Systems Management College

Program Management Course

Class 77-1

by

Edward Hudson Ely
MAJ USA

May 1977

Study Project Advisor
Mr. Edward Speca, DAC

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

SOFTWARE MANAGEMENT: A DYNAMIC APPROACH

5. TYPE OF REPORT & PERIOD COVERED

Student Project Report 77-1

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

EDWARD HUDSON ELY

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

DEFENSE SYSTEMS MANAGEMENT COLLEGE
FT. BELVOIR, VA 22060

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

DEFENSE SYSTEMS MANAGEMENT COLLEGE
FT. BELVOIR, VA 22060

12. REPORT DATE

77-1

13. NUMBER OF PAGES

42

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

UNLIMITED

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED SHEET

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE ATTACHED SHEET

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DEFENSE SYSTEMS MANAGEMENT COLLEGE

STUDY TITLE: Software Management: A Dynamic Approach

STUDY PROJECT GOALS: Determine feasibility of automated PERT/LOB applications in software management. Analyze hardware/software relationships relative to acquisition processes. Illustrate the described methodology as applied to phases of software life cycles. Evaluate concept extension with respect to future software proliferations.

STUDY REPORT ABSTRACT: The purpose of the report is to describe the concept of employing the principles of automated PERT/LOB as the foundation of a management tool used throughout software life cycles.

Project research was conducted to support the proposed conceptual methodology. Study limitations and system constraints were defined. A literature survey was conducted using the facilities and library services of the Defense Systems Management College, the US Army Computer Systems Command, and the US Army Computer Systems Support and Evaluation Agency. Discussions were conducted with software management personnel at Headquarters, Department of the Army, and within those previously cited organizations. Conversations with active program management personnel also contributed to the report.

Based upon research findings, the need for a dynamic software management system was identified. The nature and scope of the software management problem were highlighted. A conditional illustration was constructed within the framework of automated PERT/LOB principles. The concept of Interactive Computer Graphics (ICG) was then applied to the fundamental framework. The system was considered in terms of microcomputer arrays, distributed network processing, and implications of organizational policy.

Study conclusions imply that an interactive software management system is possible. Expanded use of automated PERT/LOB principles appears to be a feasible approach. From a management perspective, the hardware/software relationship during acquisition life cycles appeared inseparable. Recommendations included development and use of such an approach by program management offices.

SUBJECT DESCRIPTIONS

ADPS	Interactive Computer Graphics	Program Management
Automation	Line of Balance	Project Management
Computer Graphics	Management Systems	Software Management
Computer Software	PERT	Systems Management

NAME, RANK, SERVICE	CLASS	DATE
EDWARD HUDSON ELY, MAJ, USA	PMC 77-1	May 1977

EXECUTIVE SUMMARY

This report condenses a design concept which employs an automated system as the suggested management tool to be used throughout software life cycles.

Computer software management voids currently exist within Department of Defense (DoD). The dynamics of the Department of Defense acquisition process requires an equally dynamic management system capable of responding to such needs.

The study concludes that overcoming software management problems is feasible through expansion of automated Program Evaluation Review Technique (PERT)/Line of Balance (LOB) principles. Projection of these principles into the realm of interactive computer graphics is presented as the recommended contour for shaping future software management systems development. The methodology is seen as cost effective compared with existing approaches and is achievable within the scope of current technology.

Differences between embedded and general purpose software are reviewed. Software visibility, life cycle alignment, engineered software components, and implications of interactive computer graphs are discussed in terms of microcomputer arrays and future organizational policy.

ACKNOWLEDGEMENTS

Grateful recognitions inevitably overlook someone who has been helpful. For this reason sincerest gratitude is extended in advance to all the people who made this report possible.

Many special thanks are rendered to all members of the Army Automation Directorate, OCSA, who gave so freely of their time, but especially Mr. Sam Worthington and Mr. Art Rosenblum.

Within the US Army Computer Systems Command (USACSC), distinct appreciation is extended to MG Jack L. Hancock for his inspiration, to COL Lawrence H. Putnam for his intellectual contributions, and to COL J. E. Foster for providing valuable points of contact. Thanks are also due MAJ Jerry Thomas, Mr. John Hardwick, CPT Bob Fink, Mr. Frank Hall, Mr. Jerry Moher, and Mr. Harv Tzudiker.

At US Army Computer Systems Support and Evaluation Agency (USACSSEA) appreciation is communicated to LTC Steve Kasa, MAJ Bill Cooper, Mrs. Ruby Harney, and Mrs. Irene Kadis. Their open sharing of ideas and comments were most constructive.

Ardent gratitude is offered to all of the numerous program/project management personnel who shared their various thoughts and evaluations on the subject of software management. Thanks also go to the many folks at the US Army Management Engineering Training Activity (USAMETA), and the Mobility Equipment Research and Development Command (MERADCOM) for contributing their concepts and judgments.

Many thanks to the staff members of the libraries at the Defense Systems Management College (DSMC), USACSC, and USACSSEA for their assistance during the conduct of research and literature surveys.

A genuine note of appreciation is directed toward members of the DSMC staff and faculty for their assistance. To Mr. Edward Speca and MAJ Carlton "Rob" Roberson go much gratitude for constructive reviews and criticisms. Martha McCartney is commended for achieving such outstanding typing results from some very rough notes.

The deepest appreciation is bestowed upon my lovely wife Hoa and my understanding daughter Theresa. Their calm endurance throughout the research and writing period is what made preparation of the paper possible.

TABLE OF CONTENTS

SECTION I

INTRODUCTION

Background

For more than a century industrial machinery has evolved against the rigorous discipline of engineering standards. Computers have ostensibly exploded into existence within the past quarter century. Actually, computer machinery evolution has been controlled by the genetic discipline of engineering design. Software development has not been so fortunate. This lifeblood of computing machinery has suffered from the absence of the rigid, guided growth, and severity forced upon its hardware counterpart. As a result, today's software and its management methods have grown to appear as unguided hulks roaming through the unsuspecting world of technology seeking out trusting vict

A[illegible]view of the subject of software management is perhaps best accomplished with an approach to this discipline as though it were the quicksand of technology. Both software management and quicksand share a common characteristic. Each is a soft, shifting mass that yields easily to pressure, always changing, and always tending to pull any unsuspecting object into oblivion. Thus, the lesson extracted from the quagmire of software history and extended into the world of program management is one of constant and continued awareness.

The recent status of software conditions existing within the Department of Defense (DoD) is reflected first in the deficiency of control over increasing software expenditures. Of most concern is the continued use and dependency upon software for which validated cost control methods either do not exist or are not used. Next is the shortage of enough research and development associated with software production. This implies an obligation

on the part of all involved to exchange the art form of software for the structured form of technology. Finally, there is a dire need for meaningful upgrade in software management. This need warrants a scientific approach toward the evaluation of software. One viewpoint notes:

> "There are currently no good measures for either the software... or its practitioners. ...from a functional standpoint computer software is equivalent to hardware and must be delivered as an active system component."(21:1)[1]

Within the realm of software, the absence of disciplined and specific requirements is generating software cost overruns. This pressing void also results in system performance much below anticipated par. Support software development delays are also causing operational delays. An increasing quantity of major weapon systems dependent upon software are experiencing post-extension software support problems (18:1, 22:i). Other factors contributing to present software conditions include shortages of qualified software engineers and multiple standards for software documentation (10:3-12).

The following notion now seems appropriate to the thinking of program management teams. Commercial products reflect a corporation's changing and growing requirements. Thus, operational hardware is subject to continual modification and enhancement, "Therefore, it is even more critical that software be systematically built so it can subsequently be systematically maintained (23:24)." What is needed is a mode which makes this possible.

Overall Project Purpose

The primary purpose of this study project is to describe the concept of employing automated PERT/LOB principles as a management system throughout

---

[1]This notation will be used throughout the report for sources of quotations and major references. The first number is the source listed in the bibliography. The second number is the page(s) in the reference.

software life cycles. In addition the report discusses extension of the basic PERT/LOB idea into the more rarified atmosphere of an electronic interactive software management system. Projected implications of such a dynamic management system achieve the higher purpose of the study: to suggest a methodology for dynamic pursuit of software management in response to equally dynamic requirements for system management today and in the future. Instead of reiterating "what" should be done regarding software management, this report attempts to suggest "how" such management can be accomplished in the modern environment.

Specific Study Goals

Project study goals include determination of the feasibility of automated PERT/LOB applications in software management. Analysis of hardware/software relationships relative to acquisition processes is sought. Conceptual illustration of the described methodology as applied to life cycle phases is also desired. Evaluation of the automated software management concept is sought with respect to future program/project management software proliferations.

Scope and Limitations

As used in this report, software refers to digital computer software. Specifically, the discussion surrounds computer programs and related documentation. Such discussion does not lose sight of the broader definition of software which incorporates manuals, drawings, and so on. From the overall proposed system perspective, such software elements are active system components. Only because of the necessity to bound the scope of the study were such components eliminated from examination.

SECTION II

NATURE OF THE SOFTWARE MANAGEMENT PROBLEM

A Question of Relative Size

Annual DoD expenditures pertaining to software constitute billions of dollars (9:2, 19:i). A serious issue arises here as to whether such expenditures create, or result from, the circumstances noted earlier; namely, the lack of discipline and rigor, poor cost and schedule controls, and limited management visibility. Perhaps software management is presently out of control because it is beyond the capabilities of existing management tools. To be sure, when viewed as a combined entity, the magnitude of the software management problem is enormous. Although such a macro view is certainly essential, consideration of supporting factors is also necessary.

For too long the artificial distinction between embedded computer software (ECS) and general purpose ADP software has been allowed to continue. Embedded computer software as used here refers to software which is often described as being an integral part of a weapon system. General purpose ADP software refers to software which is often characterized or related to business-type applications. Supporting arguments usually contrast each in the following manners. Products of business-type ADP systems are simply seen as some form of an information processing effort complete with attendant application programs and other necessary software overhead. Embedded computer software is usually thought to be more specialized than ADP software. Such ECS products are often considered more difficult to employ and more elusive to evaluate except within terms of unique using systems.

Comparison of the perceived software differences between ECS and ADP identifies two often forgotten points. First, the products of ADP software are not solely the computing system nor the related file maintenance.

Rather, actual ADP products include triggered physical actions. Examples of these actions include shipments of material, summaries and projections of personnel actions, and finally, monetary flows and exchanges resulting from commercial or governmental transactions. To restate the point, the primary output of ADP software is the physical action which it causes or records, and not the computing system itself. The computing system is merely the mechanism for initiating or recording the action. Secondly, the product of embedded software may be unique to its resident weapon system only because it was engineered to be unique. Here, the computer programmer and the engineer share the common characteristic of building personal envisionment and creativity into the system without actively considering the alternative of using standardized software components to accomplish the required task. A challenge to both occupations is recognition of the important need for engineered modules and interchangeable software components which can be standardized in a manner similar to the standardization of bearings, gears, and pumps. As with ADP software, the main output of embedded software is the physical action which it causes or records, and not necessarily uniqueness of the weapon system within which it resides.

Beginning with identification of requirements, ADP software efforts can be significantly improved through application of the engineering discipline. Extraction of embedded software from its uniquely invisible resting place and elevation to the macro-interface level also appears to be a productive effort. Considered from this vantage point the differences between embedded and general purpose software become transparent. Both types of software serve similar fundamental purposes, e.g., causing or recording accomplishment of some function. Volume of input and output, the relative amount of CPU processing time, and software uniqueness are often given as justified

reasons for separation into respective classifications. Perhaps each reflects the prowess of functional disciplines rather than the true nature of software.

Software problems are sometimes seen as software errors. For example, if software does not perform according to specification, then a software error has occurred. This assumes the specification is correct, which is rarely a valid assumption. "A major source of software errors is in writing the specifications (30:2)." The point here concerns software reliability and the fact that such errors are not solely restricted to the software itself. Other contributors to software errors include interface mistakes and an improper sequence of program statements. Presently, software management's ability to satisfactorily cope with the magnitude of accurate error estimates at the functional level is, in itself, a sizeable undertaking. At higher levels of system complexity the effort appears insurmountable. Throughout the software development process from micro to macro levels, the transformation of objective requirements into software action is greatly fostered by continuously asking: What is the software supposed to be able to do?

The Need for Visibility

> "Software visibility has meaning to a PM only in terms of how it relates to his total program. This is best expressed as a function of cost, schedule, and performance. ...The key to obtaining software visibility for any PM is to elevate software out of the category of 'data' and plan for its development on a level of importance with hardware. ...There are currently no known methods for accurately measuring software development progress such that it can be effectively reported...(19)."

The foregoing observations adequately describe the need which a program manager has for software visibility. However, one might well consider that the described function is in reality a function, of a function, of a function (f(f)(f)) syndrome. Such a syndrome complicates availability of needed

software management information and puts it beyond easy retrieval of the human being.

Increasing amounts of available information and the dynamic nature of the program management decision making process are exerting significant pressures on the program manager's need for increased software visibility. Based upon the nature of software, these pressures are most likely to intensify in the future. Thus, a methodology which equals the dynamic decision making process, while fulfilling management requirements, appears warranted. Today, it is technically possible for those in program management to make today's software decisions based upon today's software management information. Further, it is possible to make such decisions without being surrounded by the walls of a paper foxhole.

Life Cycles -- Hardware/Software

If software is to be configured and managed like hardware, then a preliminary discussion of respective life cycles is appropriate. As illustrated in Figure 1a, overall hardware design and development risks are reduced as system development activities move further into the life cycle. Conceptually, software systems follow life cycle flexibility, costs, and risks similar to hardware systems. That is, a representation of the ideal software system life cycle would have the same shape as its related hardware cycle. The concept here envisions the use of engineered or standardized software components which would cause the shape of the life cycle curves to be more alike. However, actual software design and development experiences suggest that because of the inherent ease with which software is modified, just the opposite conditions result. Figure 1b depicts the commonly held viewpoint that software flexibility increases as hardware flexibility decreases.

Department of the Army Management Information System (AMIS) software

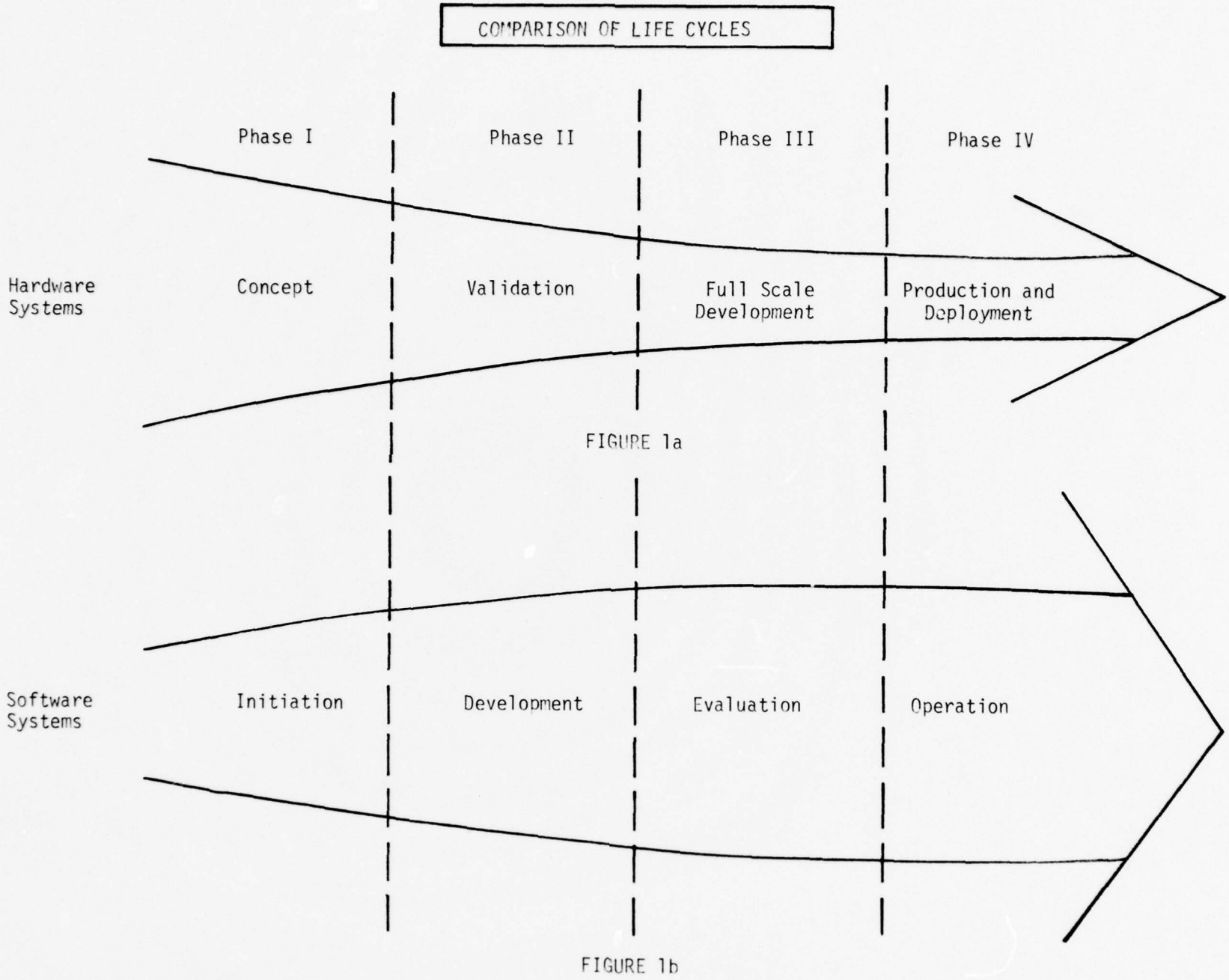


FIGURE 1a

FIGURE 1b

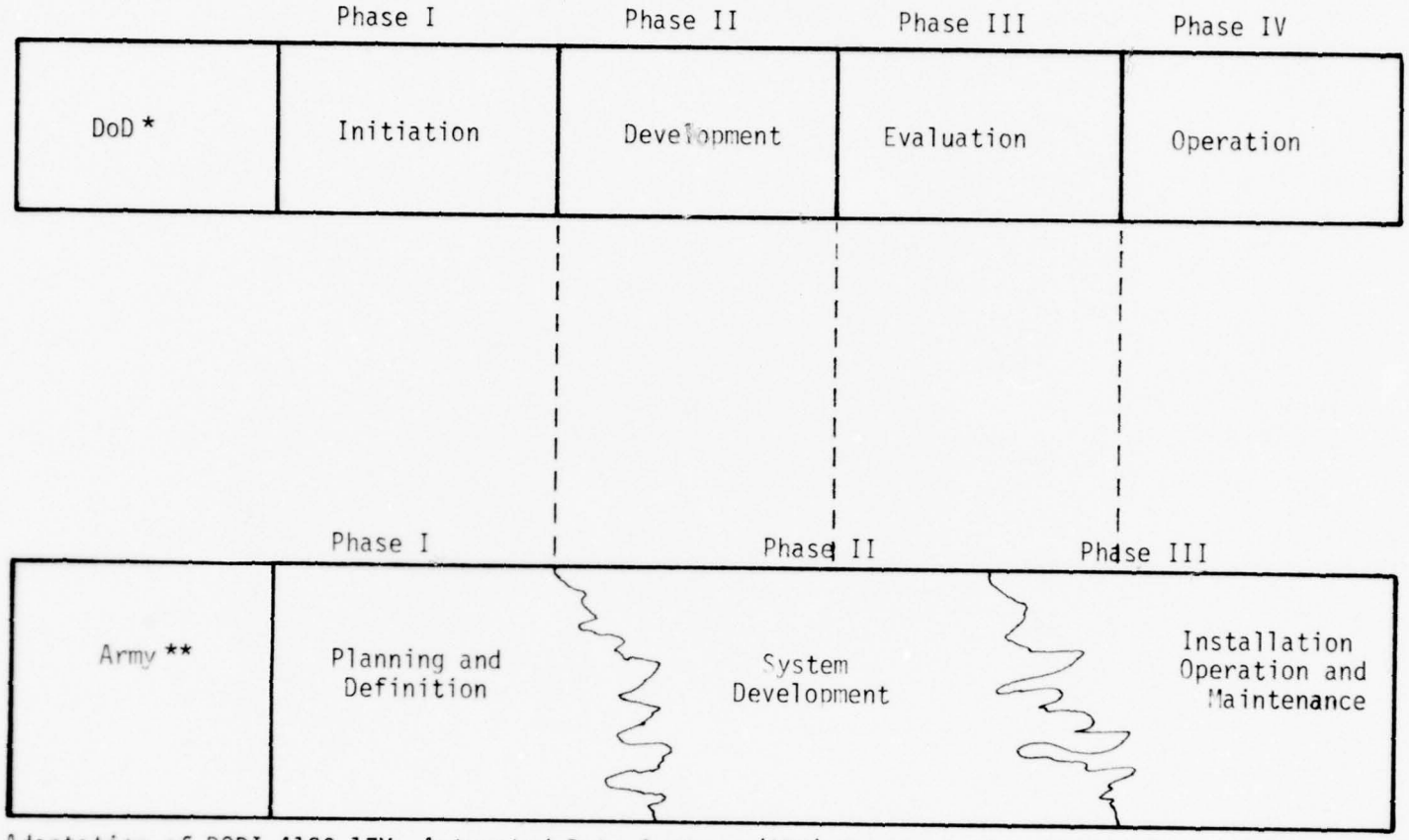
life cycle phases are compared with DoD software life cycle phases in Figure 2. Army conversion to a four-phase software life cycle similar to the DoD approach would probably provide better manpower balance throughout the software life cycle and also assist in aligning it with the computer hardware and weapon system life cycles. The key result of this action would be installation of the Evaluation Phase into the Army's software life cycle. Boundaries between the phases of the software life cycle are not usually as neatly defined as the DoD illustration would suggest. On the other hand, Army boundary lines might benefit from additional rigidity. Often, the pressures resulting from perceived software flexibility, plus the pressures to place a system into operation, are combined. This combination of forces can result in a direct transition from system development to operations with scant evaluation enroute. It would seem that program/project management personnel and software developers are obligated to resist such pressure by refusing to proliferate software systems until defined objective evaluation criteria are finalized and satisfied. It is incumbent upon software developers to say "No", rather than yield and field software known to be malignant.

It is suggested that conversion by DoD and all service components to a common software life cycle with Concept, Validation, Full Scale Development, and Production and Deployment phases would finalize the software life cycle aligrment process. Serious managerial emphasis on software development at the System Acquisition Review Council milestones could then track software development along with hardware development. For example, DSARC II would not be completed until identified and proven software objectives had been met. Obviously the purpose is to keep software visible from the outset and throughout system development.

Future System Complexities

Development of future software systems will command better management

SOFTWARE LIFE CYCLE PHASES



* Adaptation of DODI 4120.17M, Automated Data Systems (ADS) Documentation Standards Manual, Sec II Project Development Life Cycle (pp 1-4) and fig 2-01, pp 1-5, Dec 1972. (See also #37: Appendix A, List of References.)

** Illustration of AMIS life cycle as described in AR 18-1.

FIGURE 2

efforts because of increased complexity and cost requirements. In the past the use of configuration management has been widely applied to hardware, but is almost unknown to software systems applications.

Three main components of configuration management include identification of the configuration baseline, configuration control, and status accounting. These components are useful in establishment and building of the software system life cycle. Presently, configuration management standards relating to software orient mostly toward software items as component parts of major defense systems. Properly instituted, the use of configuration management applied to general purpose ADP manifests itself as a highly worthwhile undertaking.

It is often suggested that software efforts only produce one-of-a-kind items compared with hardware efforts which produce voluminous quantities of similar items. Arguments are sometimes put forth that software has no spare parts problems, and that software is only concerned with functional design. Contrary to such views are general purpose ADP systems which have worldwide distribution, and are developed and controlled from a central location. Such systems have thousands of application programs, comprised of tens of thousands of operating modules, and hundreds of thousands of lines of program code. Source programs are perhaps one-of-a-kind items, but distribution of reproduced copies of resulting object programs around the world, plus follow-on change packages, and necessary documentation does seem to create something of a voluminous software spares problem.

Certainly software's functional orientation reflects its most fundamental purpose. However, the use of an engineered software configuration eases the preparation of modules and assists in providing a definitized managerial orientation. Configuration modules are relative to the level within the system. That is, systems can have computer software module configuration items

(SMCI) composed of computer program configuration items (CPCI). Each of these moduled elements can have unique but system interrelated identity numbers, such as, computer program identification numbers (CPIN). The internals of such modules are very important, but module interfaces are absolutely critical. Modules must have guaranteed interfaces where each module receives guaranteed input and provides guaranteed output.

Other indications of future software management complexities are found in the trends and magnitude associated with recent computer machinery volume and costs. At the end of FY 76, the federal government had 9,648 computers in use. This figure represents almost 1,000 more computers than at the end of FY 75. DoD had 4,424 of the total with a combined owned/leased value of approximately $4.5 billion (40:22). Associated with each of these computers is some form of software cost, ranging from $15 - $30 per executable instruction of debugged and documented code (6:8).

SECTION III

FRAMEWORK FOR AUTOMATED VISIBILITY

Expanding Automated PERT/LOB

Overcoming the inability to accurately represent the current status of sizable software development endeavors to upper level management is perhaps the boldest challenge facing the program management team today. Software design, development, and implementation are increasingly turning into restrictive boundaries in terms of cost, schedule, and performance throughout a system life cycle. Fundamentals of Program Evaluation Review Technique (PERT) and Line of Balance (LOB) as management tools are well known, and are not presented in this report. Excellent discussions concerning PERT and LOB, including integrative views are available in several volumes (7:315-55, 33:169-174, 445-8, 453-9).

Associated use of automation with PERT/LOB is a fairly recent innovation (34:96-106). Automated PERT/LOB concepts provide a unifying and integrative management instrument with which a majority of life cycle planning and control can occur. This instrument exhibits and interrelates life cycle program elements while allowing comparison of planned with actual performance. Some modern use of automated PERT has been made in connection with program management related activities (41). Automated LOB has also been observed as a production control technique in the Army's M60 tank program. While such efforts are somewhat progressive, they labor under the burden of a batch-processing orientation. Several new efforts within the software industry have moved toward a broader networking viewpoint and are now crossing the threshold of an interactive networking capability.

An expansion of the automated PERT/LOB approach is shown in Figure 3, as an Integrated Software Management System (ISMS). This perceived enlargement illustrates an overall design concept for meeting the needs of tomorrow's managers. Such a tool is envisioned as being dynamically equal to the function for which it provides planning and control visibility. Although such a system is manually operable, application to multi-faceted projects makes the manual approach cumbersome. However, the methodology does appear applicable to both hardware and software systems management.

Linking Components of the System

System structure is depicted in Figure 3a. Such structures are often labeled as Work Breakdown, Project Breakdown, Tiered Specification, Top-Down Design, or Structured Logic. Regardless of label, the idea of subset is fundamental to the approach. The example highlights the concept of identifying levels of software subsets. These subsets are then converted into a network as illustrated in Figure 3b. A ninety-degree rotation of the structure, identification of event and activity relationships, and finally the establishment of measurement values, provide a fundamental networking arrangement.

Time measurement of network activities is typically used for schedule control purposes. It is suggested that other forms of measurement can also be calculated and equally applied to network activities. For example, optimistic and pessimistic cost estimates (C_o,C_p), early and late actual costs (C_e,C_l), optimistic and pessimistic manpower estimates (M_o,M_p), can complement usual time estimates. Critical cost and manpower paths can also be calculated in addition to normal time-related critical paths. Beginning with the system structure, identification numbers, e.g., Computer Program Identification Numbers (CPIN), are constructed according to structure level, and carried throughout the various segments of the software management system.

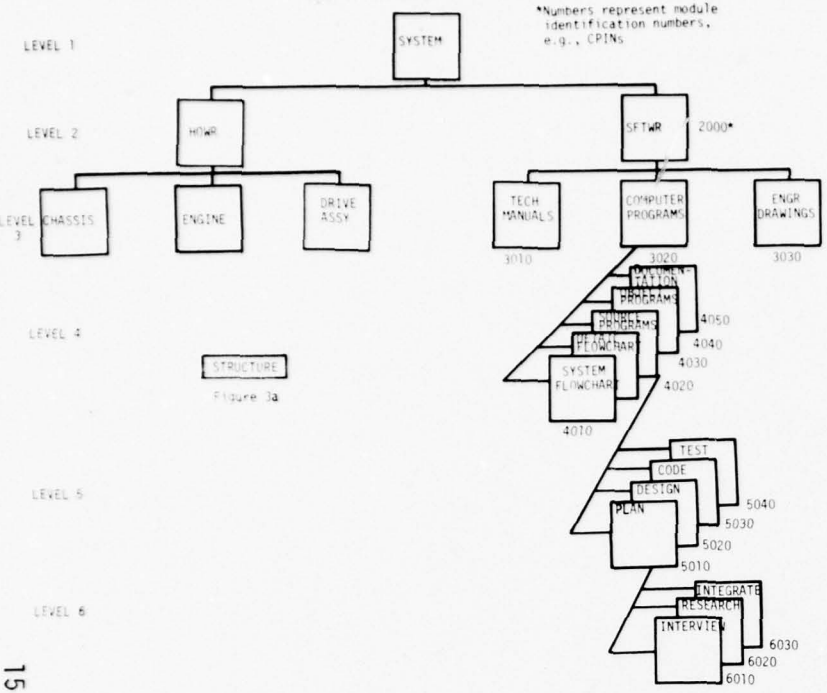


Figure 3a

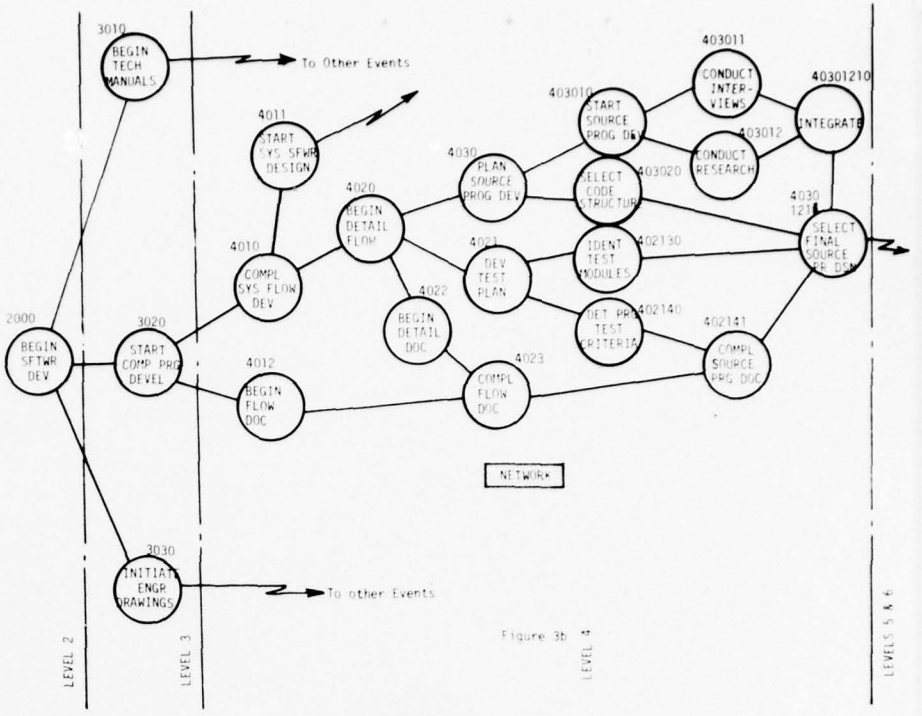


Figure 3b

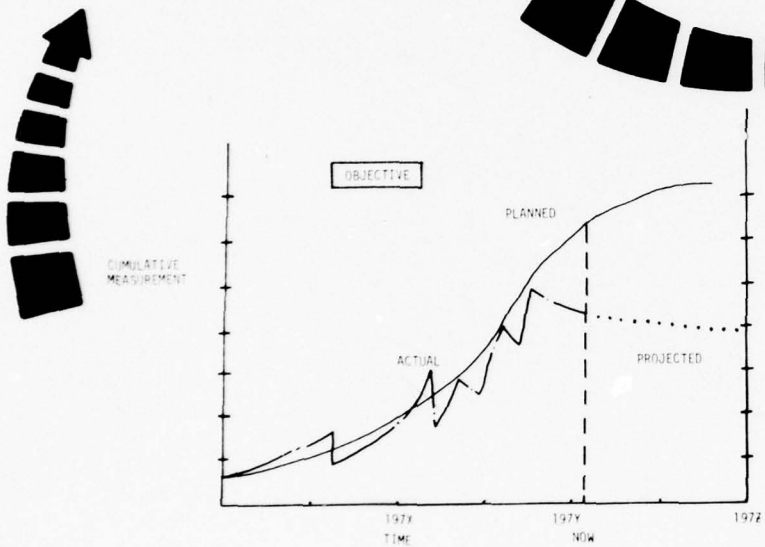


Figure 3d

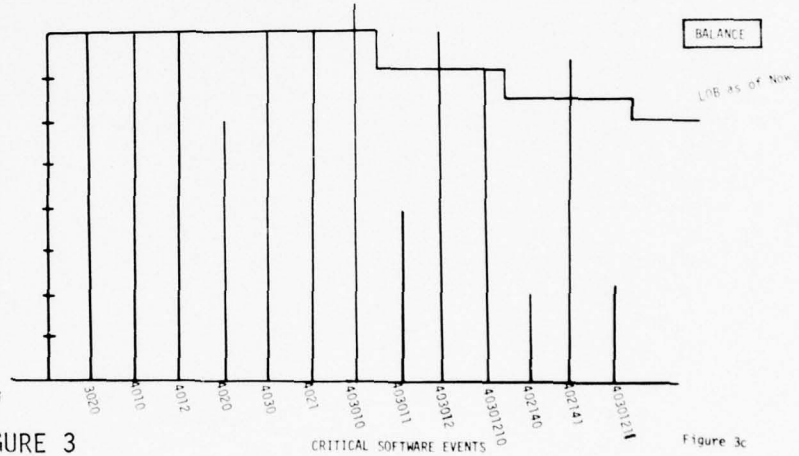


Figure 3c

FIGURE 3

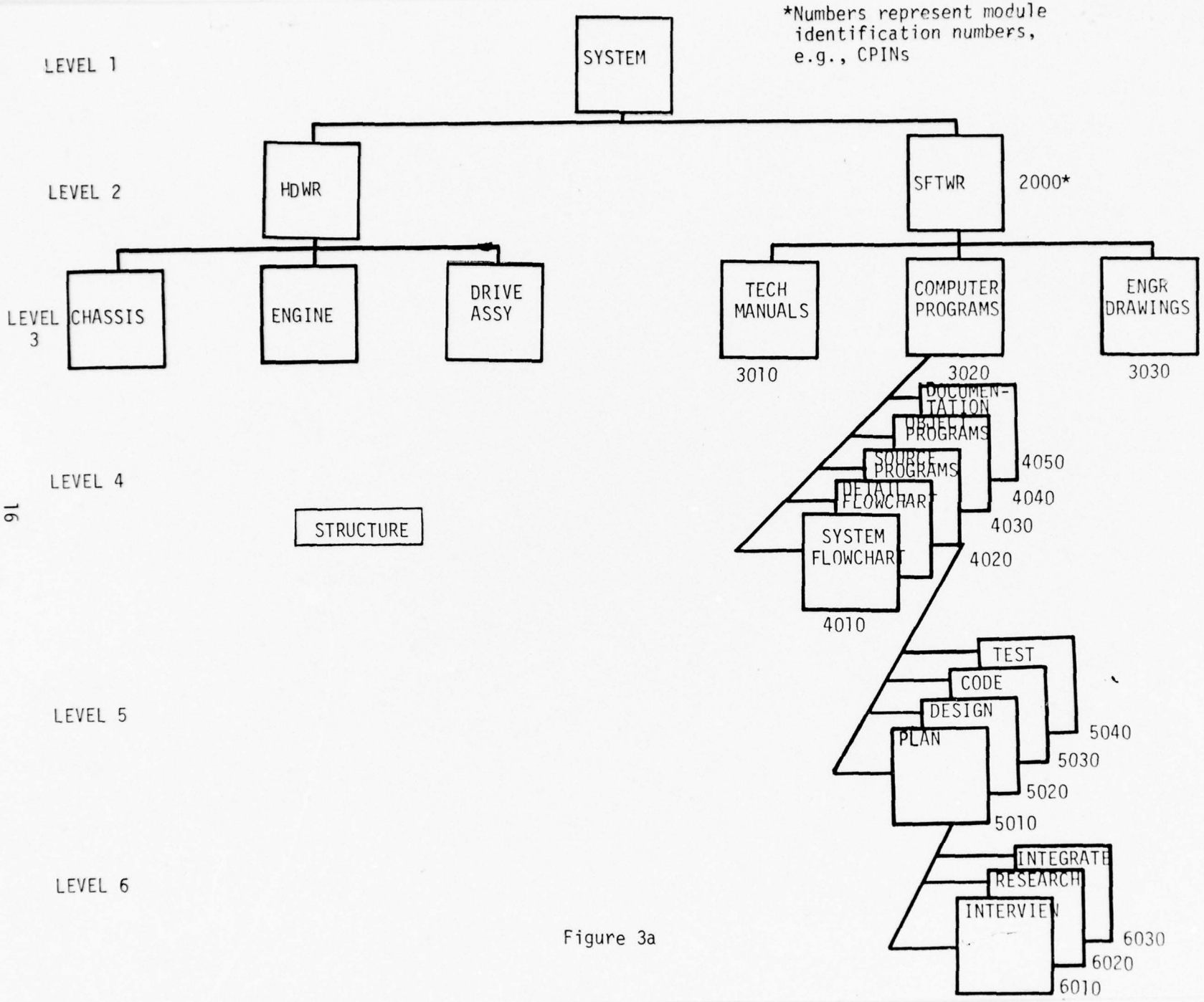

*Numbers represent module identification numbers, e.g., CPINs
LEVEL 1
LEVEL 2
LEVEL 3
LEVEL 4
LEVEL 5
LEVEL 6
SYSTEM
HDWR
SFTWR
2000*
CHASSIS
ENGINE
DRIVE ASSY
TECH MANUALS
COMPUTER PROGRAMS
ENGR DRAWINGS
3010
3020
3030
DOCUMEN-TATION
PROGRAMS
SOURCE PROGRAMS
DETAIL FLOWCHART
SYSTEM FLOWCHART
4050
4040
4030
4020
4010
TEST
CODE
DESIGN
PLAN
5040
5030
5020
5010
INTEGRATE
RESEARCH
INTERVIEW
6030
6020
6010
STRUCTURE

Figure 3a

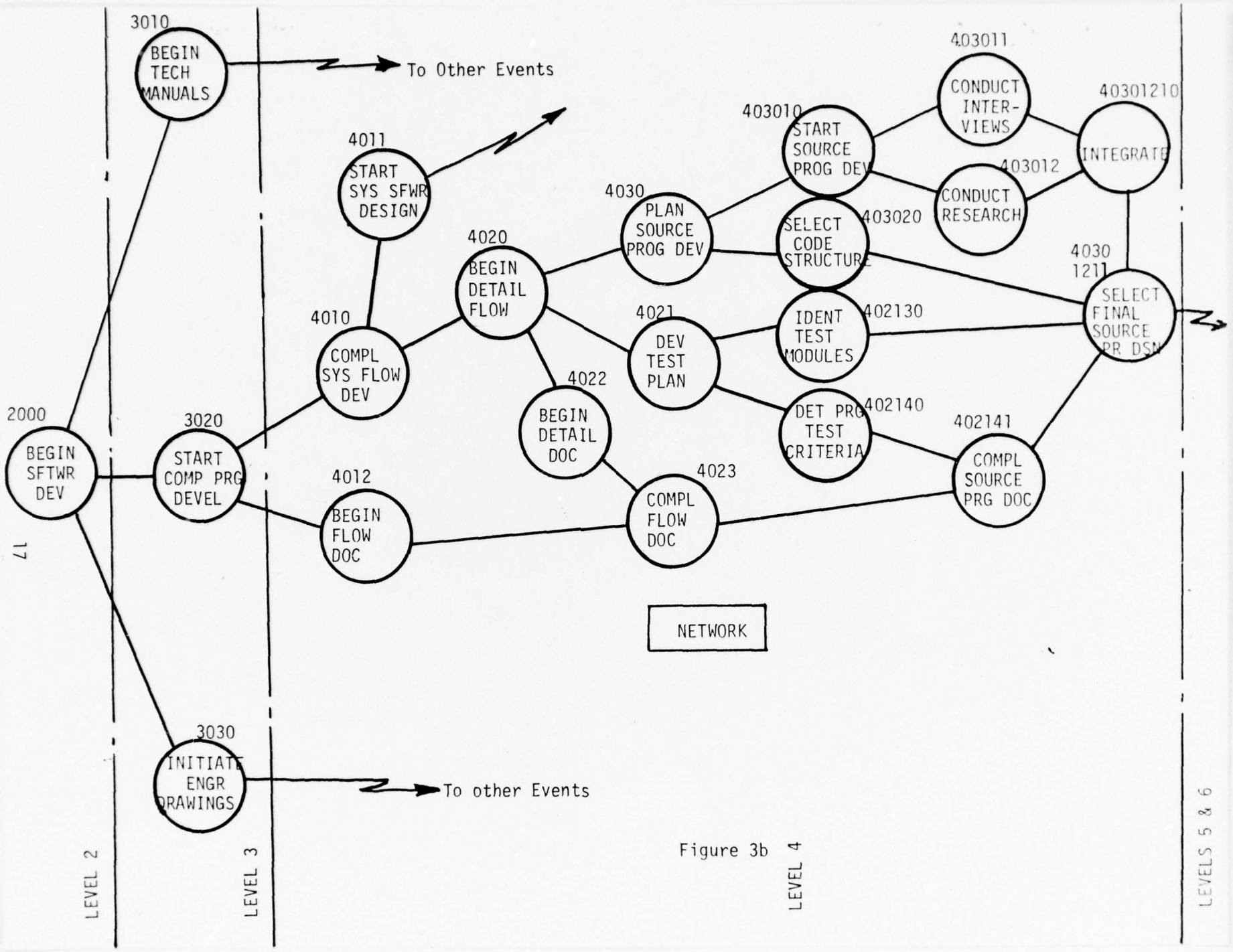

3010
BEGIN TECH MANUALS
To Other Events
4011
START SYS SFWR DESIGN
4010
COMPL SYS FLOW DEV
4020
BEGIN DETAIL FLOW
4030
PLAN SOURCE PROG DEV
4021
DEV TEST PLAN
4022
BEGIN DETAIL DOC
4023
COMPL FLOW DOC
403010
START SOURCE PROG DEV
403011
CONDUCT INTER-VIEWS
403012
CONDUCT RESEARCH
40301210
INTEGRATE
403020
SELECT CODE STRUCTURE
4030 1211
SELECT FINAL SOURCE PR DSN
402130
IDENT TEST MODULES
402140
DET PRG TEST CRITERIA
402141
COMPL SOURCE PRG DOC
2000
BEGIN SFTWR DEV
3020
START COMP PRG DEVEL
4012
BEGIN FLOW DOC
3030
INITIATE ENGR DRAWINGS
To other Events
NETWORK
LEVEL 2
LEVEL 3
LEVEL 4
LEVELS 5 & 6
17

Figure 3b

The Balance Chart displayed in Figure 3c utilizes critical events identified on the network as control points for calculation of the LOB. Such limiting events can range in hierarchy of detail from overview system summaries down to detailed levels of specification necessary for management purposes. As applied to software, these events may range from Personnel, Logistics, Financial, Production, and Development system summaries down to application program sub-routine events specified as critical for purposes of management control.

At this point it is necessary to digress momentarily and expand the sequential nature of software. If software is to be considered as hardware, then it seems to follow that the sequential relationships of software components are somewhat analogous to assemblies and sub-assemblies flowing throughout a factory floor. Moreover, it is also important to visualize the time when it will be quick and cost effective to create software modules directly onto firm circuitry materials. It is also possible to envision rapid and cost effective conversion of existing software to substrata which are more solid than existing magnetic media. Such capability makes possible software module inventories in a firm mode.

Calculation of the LOB is made against a cumulative unit of measure common to both the Objective display and the Balance display as shown in Figure 3d. One unit of measure commonly associated with industrial production is "Cumulative Units Completed". Objective Chart measurements need not be solely limited to an accumulated quantity produced. Consideration should also be given to using cumulative: "Employee Hours", "Production Hours Available", "Percent Design-to-Cost Programming Achieved", and others. If software is to be designed and managed like hardware, then it must be considered and managed as a hardware production system. Units of measure as described here can aid in such an effort.

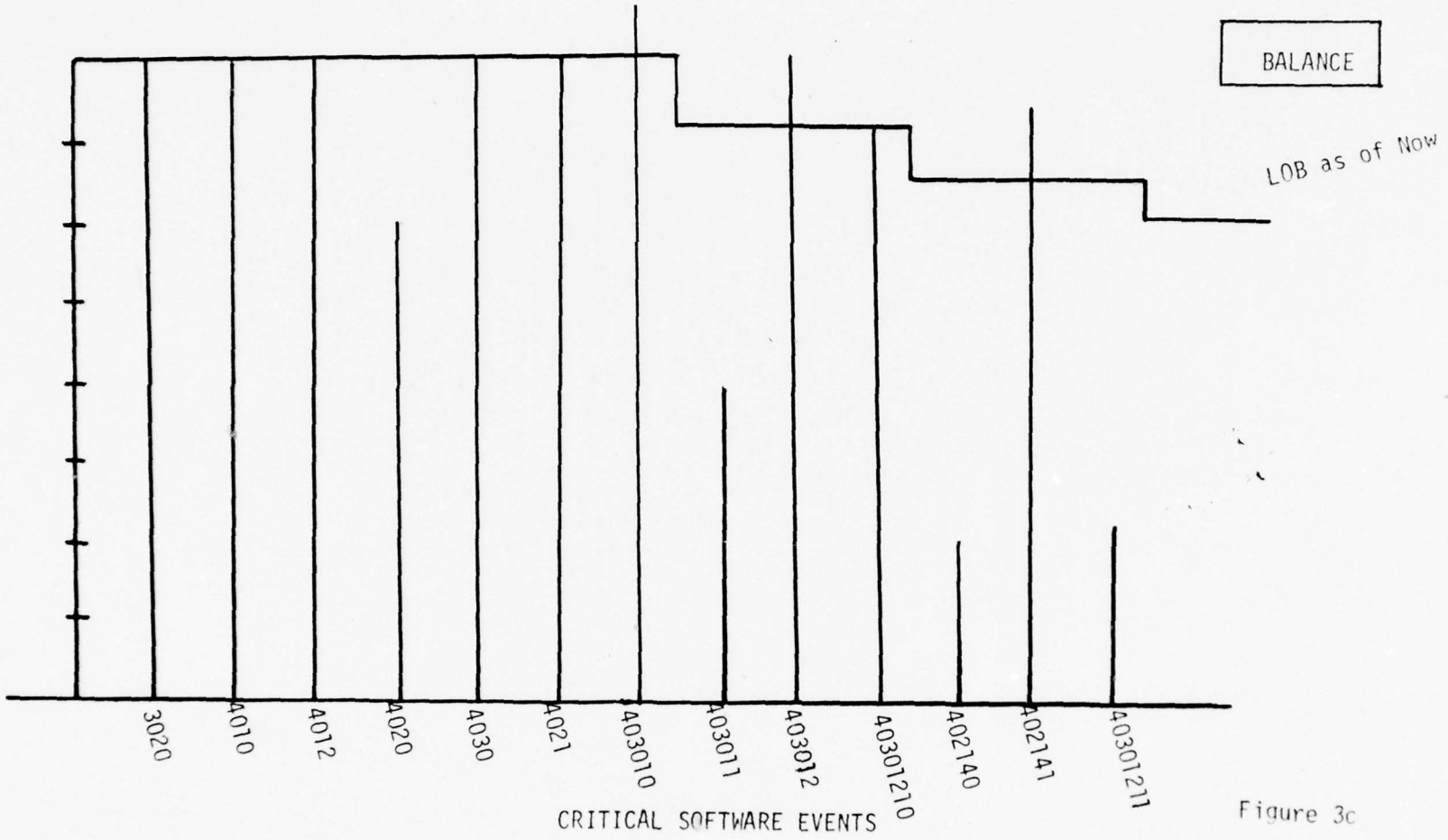

BALANCE
LOB as of Now
3020
4010
4012
4020
4030
4021
403010
403011
403012
40301210
402140
402141
40301211
CRITICAL SOFTWARE EVENTS

Figure 3c

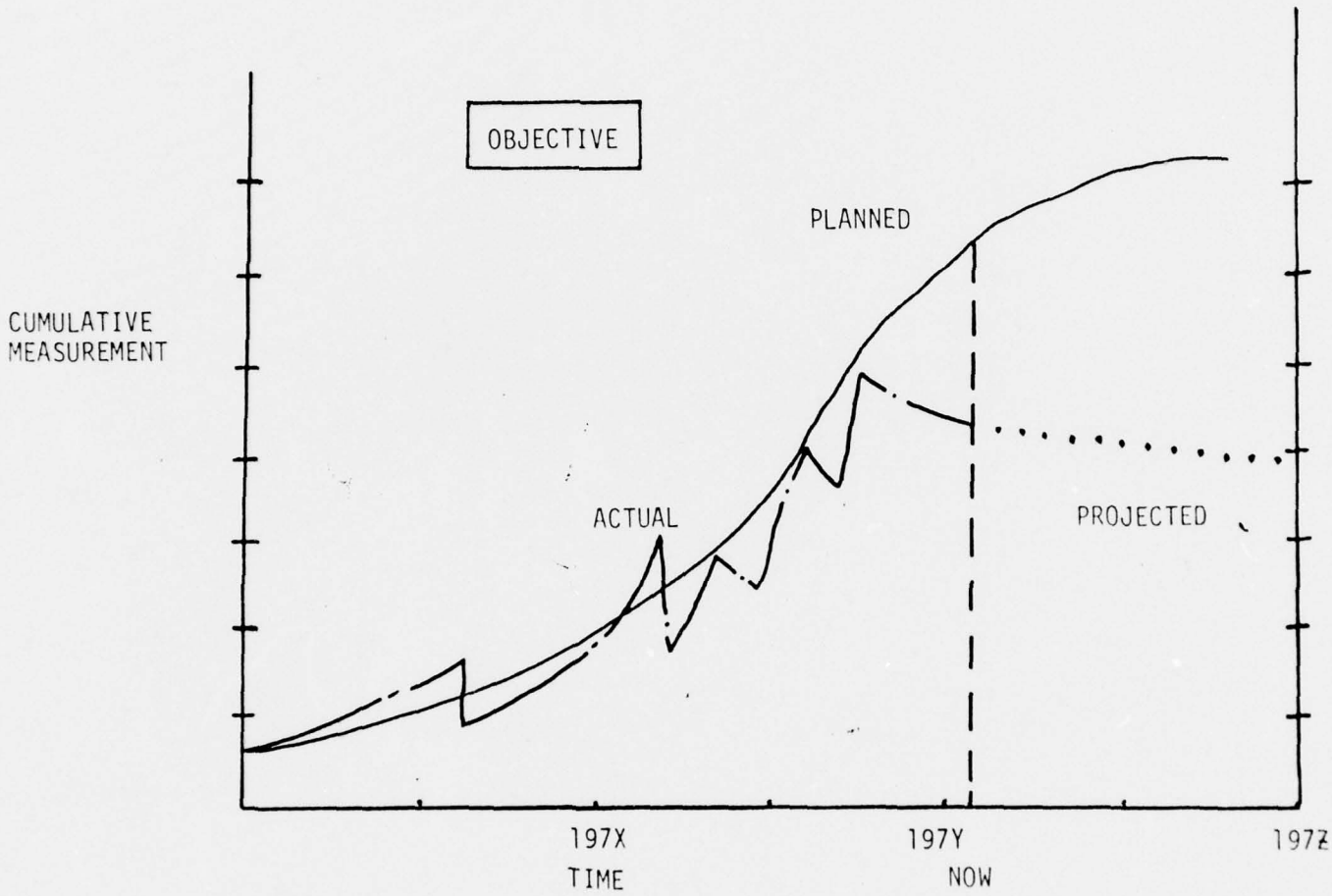

OBJECTIVE
PLANNED
CUMULATIVE
MEASUREMENT
ACTUAL
PROJECTED
197X
TIME
197Y
NOW
197Z

Figure 3d

At any point in time the planned versus actual management objectives can rapidly be evaluated in terms of deviations, variance, or incremental analysis. Required actions identified by these processes can be applied to the original structure. The advantage of using LOB as the connection between the network and the objective might be questionable to some. The value of this linkage rests in being able to summarize the interoperability of complex network occurrences and display of results in a form useful and understandable to managers in terms of established management objectives. Such visibility is not easily obtained directly from the network. Interactive actions, as previously described, flow through the Structure, Network, Balance, and Objective in a cyclic manner. Integration of software via this method can focus necessary management attention from micro to macro aspects throughout development and also correlate the requirements of system performance.

Software Life Cycle Resource Estimates

During early years of software, it was "commonly supposed that development was the main problem (28:267)". Maintenance, not development, has occupied 75 percent of ADP personnel's time over the last quarter century. Modern approaches to software life cycle management must be implemented or else maintenance requirements will seriously hinder contemporary developments in future years.

Accurate estimates of software life cycle resources are a most formidable undertaking. Experience with previous parametric software resource estimating procedures, e.g., ADP Resource Estimating Procedures (ADPREP), relied upon field-supported data bases. ADPREP suffered from the weakness of data base maintenance and the nature of parameter coefficients. Other estimating procedures range from those which are somewhat subjective (16) to more recent and

sophisticated methods grounded in quantitative objectives. While subjective estimates are better than pure conjecture, the use of macroestimating techniques such as Putnam's model (13) appear more compatible with the software management system previously described.

Use of simulation as a design evaluation tool has been suggested for the validation phase of an ECS development project (20). Broadening the concept to include simulation of the entire software life cycle is proposed via simulation of the previously discussed ISMS modules.

SECTION IV

A FUTURE ORIENTATION

Implications of ICG

Interactive Computer Graphics (ICG) systems are now being used within US business, although "Business applications using graphics are not yet very common (42:43)". The concept of business graphics also holds great promise for the program management community.

An excellent study conducted by the Defense Systems Management College (DSMC) evaluated the current status of ICG technology and included among the recommended actions,

> "That DSMC adopt and advocate the interactive computer graphics networking technique as an accepted method of program management planning and control (11)."

Some pioneering ICG applications have recently been attempted within DoD (35). However, it has also been observed that although the ICG approach is familiar to the technical sectors, there has been some reluctance on the part of managers to become involved with ICG. Part of this reluctance has been brought about by previously unacceptable ICG costs. It is also possible to speculate that some reluctance has been fostered by prior distrust of computers and former unfilled promises associated with automation. The time has come to think well beyond the limited punch card. The time has come to think in terms of electronic and interactive management images.

Use of ICG is advocated here as the methodology for design, development, production, and control of future software program management efforts. Figure 4 illustrates extension of the ISMS concept put forth earlier. It is envisioned that incorporation of ICG technology thus creates an Interactive Software Management System. Numerous significant advantages accrue from such interactive capability:

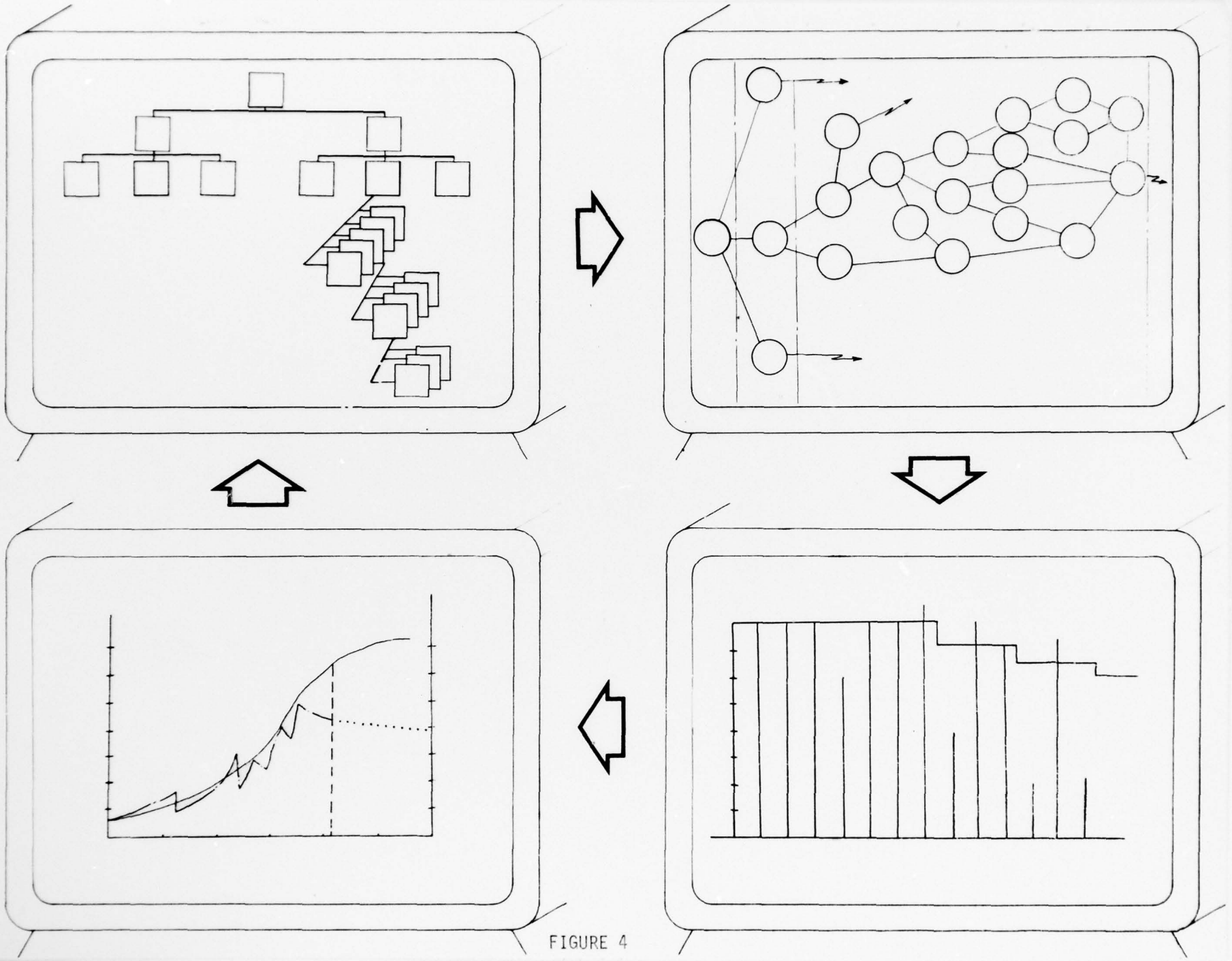


FIGURE 4

-- Instant availability of current information so vital to program management decision making.

-- "What if" capability providing opportunity to view the impact of decisions as they ripple throughout the system.

-- Ease of system use and data base update via "menuing" and "windowing" techniques.

-- Focusing on specific segments, or single items, within the Structure, Network, Balance, or Objective displays.

-- Rapid electronic remake, redraw, and edit capability for Structure, Network, Balance, and Objective displays.

-- Soft copy displays with selected hard copy option.

-- Electronic roll up of information for purposes of executive display or as input to an electronic version of a macroestimating model.

-- On-line analysis and programming.

-- Interactive Computer Assisted Instruction (CAI) and Computer Managed Instruction (CMI) for managerial tutoring purposes.

Advancement into the interactive hemisphere of software management is now technically feasible. Such moves are rapidly becoming cost effective in light of declining hardware costs and the limited capabilities, and doubtful benefits, resulting from existing approaches to software management.

The requirement for dynamic software management presently exists, a conceptual design approach has been presented, and available interactive technology has already been applied to similar management systems employing business graphics. The use of ICG to develop an ISMS is recommended. All that appears to remain is managerial commitment to development and proliferation of such a system.

Microcomputer Arrays and Distributed Visibility

Widening the immediate ICG horizon reveals the world of microprocessor technology. Today the significance of microprocessors is reflected in very low priced, highly powerful computer systems. Such systems are sometimes physically small and rugged, but possess capabilities far beyond the

computer systems which they replace. As a result of microprocessor technology, operation costs can be significantly reduced. Also, quantities of system components are lessened using microprocessor technology (29).

Sizable program management ventures are certainly better able to budget for interactive software management systems than are smaller programs. So, although smaller programs experience a need for interactive software management, they are limited in the amount of funding available for such excursions. Overcoming this deficiency becomes possible through use of arrayed microcomputers, centrally located, employing distributed networks, and utilizing time-shared interactive options. This extension impressively reduces associated ISMS costs, while placing ISMS capability within affordable reach of any program management team. Ultimately, arrayed microcomputers could provide an interactive total system data base. Such a base would provide intraoperability within the system compendium and perhaps interoperability with other such systems. The system would also provide selected DoD executives, service component officials, and the program management team with the capability and opportunity to rapidly gain macro visibility of system life cycles for defense decision making purposes.

Some Organizational Considerations

The necessity for intensive management over software system design, development and production has become evident. Until recently, much DoD organizational emphasis has been placed on controlling the procedures involved with acquisition of computer machinery (24, 26). Attempts by Department of the Army to overlay program management concepts and principles onto computer resource acquisitions are also in existence (14, 15).

At issue here is the distinction between organization emphasis upon software and the machinery emphasis. Should the discrimination be machinery

first or software first? What is the comprehensive automation plan? A partial approach toward resolution of this dichotomy is noted in recent Department of the Army action at the policy level (12). This action attempts to unify Army automation under the centralized guidance of the Director of Army Automation (DAA). As described in the DAA charter, "The DAA's authority and responsibility for Army automation extends to all computer resources (12:9)".

> "The term 'Army automation' is used in a restrictive sense to describe computer resources that are used by automated systems whose primary purpose is to process data or information in support of management or mission functions; automated systems configured to operate in specialized environments; and combat weapons systems which have embedded computers (12:iii)."

The notion of interactive software management systems, coupled with clarification of automation policy, are put forth as two broad attempts striving to overcome part of the DoD software management problem.

Missing is a solid third proposal to advance defense system acquisition management via combined interactive software/interactive hardware program management systems. Such combined management represents the essense of defense systems acquisition visibility. Software and hardware are mirror images. They are destined to be interactively managed.

SECTION V

SUMMARY

Conclusions

The purpose of this report has been to describe a potential design concept for overcoming the DoD software management problem. Software has not evolved as an engineer-controlled discipline. Software management is often not conducted in a disciplined manner. There is a shortage of qualified software management talent. These circumstances have contributed to uncontrolled software cost increases and reduced system performance.

A need exists to convert software and its management from an art to a technological basis. This suggests a need for a method of systematically building and maintaining future software. Automated PERT/LOB is described as the foundation of a management system for addressing several of the stated software management needs throughout system life cycles.

The study also projects automated PERT/LOB into a dynamic interactive software management system. The ECS versus ADP software distinction is challenged in the context of the nature of software, its size, and intended function. Importance of the need for software visibility is highlighted. The report also suggests that an alignment of life cycle phases between software and hardware would be worthwhile. It also recommends that identified and defined software objectives be enforced as achieved requirements at life cycle development milestones. Future systems are foreseen as more complex and dynamic in their need for progressive management methods. The study considers use of software modules as standardized software components similar to standardized hardware components.

In terms of study goals the use of automated PERT/LOB as the foundation for a software management system appears feasible. Hardware/software

relationships during acquisition life cycles were analyzed and determined to be inseparable from a management perspective. The ISMS methodology was described in terms of integrated hardware/software acquisition processes. Application of ICG technology, use of microcomputer arrays, and future organizational considerations were also suggested.

Recommendations

Expand the concepts of automated PERT/LOB into an integrated framework for management review. Link ISMS components as an interactive system. Employ modern software resource macroestimating methods. Overlay ICG for purposes of software design, development, production, and control onto ISMS. Extend the methodology into arrayed microcomputer systems for multiple program management team usage. Modify organizational policy to equally emphasize software and hardware. Interactively manage hardware and software as a combined entity by developing such systems for use by Program Management Offices.

LIST OF ILLUSTRATIONS

LIST OF DEFINITIONS

The terms listed below are defined for use within the limited context and purpose of this report. Numbers in parentheses relate to source.

ADP (Automatic Data Processing) - This acronym pertains to equipment such as EAM (Electronic Accounting Machines) and EDP (Electronic Data Processing) equipment units, or systems. Data processing performed by a system of electronic or electrical machines so interconnected and interacting as to reduce to a minimum the need for human assistance or intervention. (1)

ADPREP (ADP Resource Estimating Procedure) - A parametric technique used for estimating ADP resources. The technique is explained in US Army Technical Bulletin 18-19-1.

Batch-Processing - A technique by which items to be processed must be coded and collected into groups prior to processing. A systems approach to processing where a number of similar input items are grouped for processing during the same machine run. (1)

CAI (Computer-Aided Instruction) or (Computer-Assisted Instruction) - An educational concept which places the student in a conversational mode with a computer which has a preprogrammed study plan. The programmed course selects the next topic or phase of study according to previous responses from the student, allowing each student to progress at a pace directly related to his learning capability. (1)

CMI (Computer Managed Instruction) - A concept which employs a computer in a conversational mode for the purposes of managing course curriculum or presentation of instruction. (5)

CPCI (Computer Program Configuration Item) - Configuration Item (CI) in terms of computer program software, where CI is an aggregation of software or any of its discrete portions which satisfies an end use function and is designated for configuration management. In the broader sense can include an aggregation of hardware and software. (3)

CPIN (Computer Program Identification Number) - A unique identification number associated with a computer program for purposes of CI control. (3)

ECS (Embedded Computer Software) - Computer software which is normally considered an integral part of a weapon system. Often distinguished from general purpose ADP. (5) (See also #9, List of References)

Function - Refers to a mathematical correspondence that assigns exactly one element of one set to each element of the same or another set. A sequential function. (4)

Hardware - In the automation sense refers to the mechanical, magnetic, electrical and electronic devices or components of a computer. The electric, electronic, and mechanical equipment used for processing data consisting of cabinets, racks, tubes, transistors, wires, motors, and such. Any piece of automatic data processing equipment. (Slang) Otherwise, the term is used as a reference to machinery or weapon systems. (1)

Interactive - a concept wherein there is interaction between a human being and a computer. As applied to computer graphics the term relates to the idea of placing information into and receiving information from the computer via terminal display. (5)

Life Cycle - Same as System Life Cycle - The phases through which a system passes from conception to disposition. (2)

LOB (Line of Balance) - A planning system, normally described in terms of production, which is used to schedule key events with respect to project completion. Such information is related to management objectives for comparison of actual with scheduled achievement. The system generates a line of balance to display revised requirements for keeping on schedule, or within cost and manpower limits. It is used primarily for planning and control. (2)

Menuing - The idea of visually presenting electronic listings or "menus" as displays on computer terminals and allowing selection or manipulation of various entries. (5)

Microcomputer - A general term referring to a complete microminiature computing system consisting of hardware and software. Main processors are made of semiconducter integrated circuits. In function and structure somewhat similar to a minicomputer, with main microcomputer differences being lower price, size, speed of execution, and computing power. The concept of microcomputer arrays visualizes such arrays comprised mainly of microprocessor memory and circuitry. (1)

Microprocessor - The semiconductor Central Processing Unit (CPU) and one of the principle components of the microcomputer. The elements of microprocessors are frequently contained on a single "chip" or within the same package, but sometimes are distributed over several separate chips. The term "chip" refers to Large Scale Integrated (LSI) chip circuits. A concept which employs microminiaturized circuitry, where an entire computer processor or processor segment, is contained on a single chip. (1)

MIS (Management Information System) - A specific data processing system that is designed to furnish management and supervisory personnel with information consisting of data that are desired, and which are fresh or with real-time speed. A communications process in which data are recorded and processed for operational purposes. The problems are isolated for higher-level decision making and information is fed back to top management to reflect the progress or lack of progress made in achieving major objectives. (1)

Optimistic - An estimate made at a particular point in time of the most favorable, likely, or best outcome, concerning a future event or activity. An expected favorable outcome. (4)

PERT (Program Evaluation and Review Technique) - A set of principles, methods, and techniques for network planning of objective-oriented work thereby establishing a basis for effective scheduling, costing, controlling and replanning in the management of programs. It employs: A product-oriented work breakdown structure, beginning with these objectives subdivided into

successively smaller end-items. A flow plan consisting of all the activities and events that must be completed or accomplished to reach the program objectives, showing their planned sequence of accomplishment, interdependencies, and interrelationships. Elapsed time estimates and identification of critical (or limiting) paths in the networks. A schedule which attempts to balance the objectives, the network flow plan, and resources availability. Analysis of the interrelated networks, schedules, and slack values as a basis for continuous evaluation of program status, forecast of overruns, and the identification of problem areas in time for management to take corrective action. (2)

Pessimistic - An estimate made at a particular point in time of the most unfavorable, unlikely, or worst outcome concerning a future event or activity. An expected unfavorable outcome. (4)

Program/Project Management - A concept for the technical, business, and administrative management of specified development/acquisition programs based on the use of designated, centralized management authority. This authority is responsible for planning, organizing, directing, and controlling all phases of research, development, initial procurement, production, distribution, and logistical support for the purpose of providing a balanced program to accomplish the stated program objectives. It is also responsible for assuring that planning is accomplished and action is implemented by the organizations responsible for the complementary functions of evaluation, logistic support, personnel, training, operational testing, activation, and employment. Program management implies a level of centralized management greater in scope and responsibility than project management. A program management office may be superimposed over one or more project management offices. (2)

Software - The internal programs or routines professionally prepared to simplify programming and computer operations. These routines permit the programmer to use his own language (English) or mathematics (Algebra) in communicating with the computer. Various programming aids that are frequently supplied by the manufacturers to facilitate the purchaser's efficient operation of the equipment. Such software items include various assemblers, generators, subroutine libraries, compilers, operating systems, and industry-application programs. In the restricted sense computer software is a combination of associated computer programs and computer data required to enable the computer equipment to perform computational or control functions. In the broader sense software also includes such items as documentation, manuals, and engineering drawings. (1)

Structured Logic - The process or application of structured thought. An application of the structure theorem where an entity is a complex configuration of elements, parts, or constituents in a specifically configured organization or arrangement. Also includes the interrelationships of parts, or the principle of organization in a complex entity. The concept of system, sub-system, sub-sub-system ad infinitum, where each system level interfaces with respective subordinate, ordinate, and super-ordinate levels via guaranteed interfaces. Application of such thought is often observed in the conduct of structured analysis and during design and development of structured computer programs. (4)

LIST OF DEFINITIONS (Continued)

Top Down Design - The concept of hierarchical design encompassing tiered levels, specifications, and modules subordinate to the overall or total system level. This design approach is observed in tree diagrams, breakdown structures, and top down structured programming.

Windowing - The idea of being able to electronically frame a segment of an image on a computer terminal video display and modify the segment by scoping in on or expanding the "windowed" frame. The ability to enlarge, reduce, consolidate, or change a particular segment of an automated data base from a display terminal. As applied to a network, the events and activities are electronically consolidated, expanded, and modified to enhance satisfaction of managerial requirements.

Definitions were extracted, condensed, and modified from the following sources, or specifically defined for use within this report, and may not be universally applicable:

1. Computer Dictionary, Sippl, Charles J. and Sippl, Charles P., Howard W. Sams & Co., Inc., Indianapolis, 1974-1976 editions.

2. Defense and Aerospace Glossary for Project Management, J. Ronald Fox, Hawthorne Publishing House, Washington, D. C., 1970.

3. DODI 5010.21, "Configuration Management Implementation Guidance," August 6, 1968, AFR 800-14, MIL-STD-1521 (USAF). (See also #2, List of References)

4. The American Heritage Dictionary of the English Language, American Heritage Publishing Company, Inc. and Houghton Mifflin Co., Boston, 1969.

5. As defined by author.

BIBLIOGRAPHY/LIST OF REFERENCES

1. Anderson, Lee A., "Software Management: Applying Hardware Disciplines to Software Management," Study Project Report PMC 76-2, Defense Systems Management College (DSMC), Fort Belvoir, Virginia, November 1976.

2. Anway, Mark D., "Configuration Management for the Development of Computer Systems," Study Project Report, Individual Study Program (ISP), DSMC, PMC 76-2, November 1976.

3. Archibald, Russell D., Managing High-Technology Programs and Projects, New York: John Wiley & Sons, 1976.

4. Barclay, Douglas H., "The Project Manager and Systems Analysis," ISP, DSMC, PMC 74-1, May 1974.

5. Benjamin, Robert I., Control of the Information System Development Cycle, New York: Wiley-Interscience Division, John Wiley & Sons, 1971.

6. Bucciarelli, Marco A., "Technical Performance Measurement for Computer Software Development Programs," ISP, DSMC, PMC 74-1, May 1974.

7. Cleland, David I. and King, William R., Systems, Organization, Analysis, Management: A Book of Readings, New York: McGraw-Hill Book Company, 1969, pp 315-355.

8. Cleland, David I. and King, William R., Systems Analysis and Project Management, New York: McGraw-Hill Book Company, 2d Edition, 1975.

9. De Roze, Barry C., "An Introspective Analysis of DoD Weapon System Software Management," Defense Management Journal, Vol. 11, No. 4, October 1975, pp 2-7.

10. Defense Systems Management College Report on DSMC Workshop, 16-17 June 1976, "Management of Software Acquisition for Embedded Computer Systems", DSMC, Fort Belvoir, VA.

11. Defense Systems Management College Research Report by Joseph E. Callahan, Carlton F. Roberson, and George H. Perino, Jr., "Interactive Computer Graphics: A Responsive Planning and Control Tool for DOD Program Management," January 1977, n.p.

12. Department of the Army, OCSA, "Automation Management Study (Director of Army Automation)," 25 February 1977.

13. Department of the Army Pamphlet No. 18-8, "Management Information Systems: A Software-Resource Macroestimating Procedure," February 1977.

14. Department of the Army Regulation 18-1, "Management Information Systems Policies, Objectives, Procedures and Responsibilities," 22 March 1976.

15. Department of the Army, USACSC Regulation 18-4, "Management Information Systems Project Officers," 5 March 1975.

16. Department of the Army, USACSCSGL Memorandum 18-1, "Chapter 5: SCR Estimating Procedures," 16 December 1976.

17. Department of Defense Directive 5000.29, "Management of Computer Resources in Major Defense Systems," 26 April 1976.

18. Department of the Navy, Naval Sea Systems Command, "Product Quality Assurance for Shipboard Installed Computer Programs," Pathway Program Management Plan, Vol. 1, 31 October 1976.

19. Driscoll, Alan J., "Software Visibility for the Program Manager," ISP, DSMC, PMC 76-2, November 1976.

20. Feingold, Robert S., "Computer System Simulation: A Design Evaluation Tool," ISP, DSMC, PMC 76-1, May 1976.

21. Gansler, Jacques S., "Comment," Defense Management Journal, Vol. 11, No. 4, October 1975.

22. General Accounting Office, *Report to the Congress by The Comptroller* General of the United States, "Problems In Developing The Advanced Logistics System," Department of the Air Force, 17 June 1976.

23. Goetz, Martin A., "Seen as Machine, Software's Image May Improve," Computerworld, 13 December 1976.

24. Haars, Neil W., "Minicomputer System Acquisition Within DoD," ISP, DSMC, PMC 76-2, November 1976.

25. Manley, John H., "Embedded Computer System Software Reliability," Defense Management Journal, Vol. 11, No. 4, October 1975, pp 13-18.

26. Marr, Francis C., "General Purpose Computer Acquisition," ISP, DSMC, PMC 76-2, November 1976.

27. McCarthy, Rita, "Applying the Technique of Configuration Management to Software," Defense Management Journal, Vol. 11, No. 4, October 1975, pp 23-28.

28. Mills, Harlan D., "Software Development," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, December 1976, pp 265-273.

29. Mitchell, Stuart G., "Microprocessor Technology for Managers," ISP, DSMC, PMC 76-1, May 1976.

30. Myers, Glenford J., "Software Errors -- Some Explanations," Software World, Vol. 8, No. 1, November 1976, pp 2-8.

31. Parke, Robert H., "Procurement of ADPE in the Army: An Evaluation," ISP, DSMC, PMC 76-1, May 1976.

32. Perino, George H. Jr., "Interactive Computer Graphics: Applications for Program Management," ISP, DSMC, PMC 76-2, November 1976.

33. Riggs, James L., Production Systems: Planning, Analysis, and Control, New York: John Wiley & Sons, 1970, pp 169-174, 445-448, 453-459.

34. Schoderbek, Peter P. and Digman, Lester A., "Third Generation PERT/LOB," Harvard Business Review, September-October 1967.

35. Seay, Douglas C., "Use of an Interactive Computer Graphics Model in Army Project Planning and Control," ISP, DSMC, PMC 76-2, October 1976.

36. Sippl, Charles J. and Sippl, Charles P., Computer Dictionary, Indianapolis: Howard W. Sams & Company, Inc., 1974.

37. Smith, Ronald L., "Estimating Software Project Resource Requirements," Final Report, Structured Programming Series, Vol XI (Maryland: IBM, Corporation Federal Systems Center under USAF(RADC) Contract F30602-74-C-0186, and co-sponsored by USACSC), 22 January 1975.

38. Williams, James K., "Software Management for Shipboard Computer Systems," ISP, DSMC, PMC 76-2, November 1976.

39. Zempolich, Bernard A., "Effective Software Management Requires Consideration of Many Factors," Defense Management Journal, Vol. 11, No. 4, October 1975, pp 8-12.

40. "Inside Info", Infosystems, Apr 77, Vol. 24, No. 4.

41. "Study of Automated Program Management Techniques", USAMETA, Rock Island, Illinois, November 1976.

42. "Graphic Systems for Business", Infosystems, April 1977, Vol. 24, No. 4.